

# **Results from the *read-15-3* test**

**Adam Lyon, August 23, 2005**

## **1 Introduction**

A SAM test was performed to determine the rate of file deliveries possible on a test CAF system.

Andrew made a change to the station in the middle of the test that made significant improvement!

## Appendix A R Session

R is an open source statistical analysis software package that allows for very easy analysis of data in databases and text files. I wrote a "notebook" style package that allows one to use R from within Microsoft Word. Below is the notebook providing all of the code and results for this document.

### A.1 Connect to R and initialize

Connect to R running locally on my laptop.

```
<R0> #connect port 6101 timeout 20
R is using work directory /Users/adam/work/projects/cdfSamTests/read-
15-3
```

Set up graphics

```
<R1> library(lattice)
```

```
<R2> trellis.par.set(col.whitebg())
```

```
<R3> fontsize = trellis.par.get("fontsize"); fontsize$text=16 ;
      fontsize$points=6 ; trellis.par.set("fontsize", fontsize)
```

I have a helper function that makes putting graphics into Word easy.

```
<R4> mp
function (plotExpr, file, height = 7, width = 7, res = 72 * 3)
{
  bitmap(file, "pngalpha", height = height, width = width,
         res = res, pointsize = 10)
  r = eval(plotExpr)
  if (class(r) == "trellis")
    print(r)
  invisible(dev.off())
}
<environment: namespace:RemoteRSOAP>
```

### A.2 Running job output

Doug's python script loops over getting the next file, waiting thirty seconds to simulate processing, and then releasing the file. A log file records the time of the get next file, the duration of the command, the pnfs name of the file, and the time and duration of the release file command.

Let's read this information into R.

```

<R5> d = read.table("out.log", header=T)

<R6> d[1:5,]
   job segment   getDate   getTime   getDur fileNum
1 10280        1 2005-08-23 14:01:51 400.155      1
2 10280        1 2005-08-23 14:08:46 500.057      2
3 10280        1 2005-08-23 14:17:22 225.075      3
4 10280        1 2005-08-23 14:21:22 680.100      4
5 10280        1 2005-08-23 14:32:57 460.094      5

pnfs
1
dcap://cdfdca2.fnal.gov:25149/pnfs/fnal.gov/usr/cdfen/filesets/GJ/GJ22/
GJ2270/GJ2270.0/xd025982.029bbhd0
2
dcap://cdfdca2.fnal.gov:25149/pnfs/fnal.gov/usr/cdfen/filesets/GJ/GJ22/
GJ2270/GJ2270.0/xd025a0e.00e7bhd0
3
dcap://cdfdca2.fnal.gov:25149/pnfs/fnal.gov/usr/cdfen/filesets/GJ/GJ22/
GJ2270/GJ2270.0/xd0259a7.001ebhd0
4
dcap://cdfdca2.fnal.gov:25149/pnfs/fnal.gov/usr/cdfen/filesets/GJ/GJ22/
GJ2270/GJ2270.0/xd0259a5.01d0bhd0
5
dcap://cdfdca2.fnal.gov:25149/pnfs/fnal.gov/usr/cdfen/filesets/GJ/GJ22/
GJ2275/GJ2275.0/xd025b54.00c8bhd0
    relDate   relTime   relDur
1 2005-08-23 14:08:46  0.181
2 2005-08-23 14:17:21  0.112
3 2005-08-23 14:21:22  0.272
4 2005-08-23 14:32:57  0.327
5 2005-08-23 14:40:52  0.115

```

Convert dates and times into POSIX times that R can deal with...

```
<R7> d$getDateTime = paste(d$getDate, d$getTime)
```

```
<R8> d$relDateTime = paste(d$relDate, d$relTime)
```

```
<R9> d$getPTime = as.POSIXct( strptime( d$getDateTime, "%Y-%m-%d
%H:%M:%S" ) )
```

```
<R10> d$relPTime = as.POSIXct( strptime( d$relDateTime, "%Y-%m-%d
%H:%M:%S" ) )
```

We can make the delivery time from the get time plus the duration

```
<R11> d$delPTime = d$getPTime + d$getDur
```

What is the range of the test?

```
<R12> testEdges = c(min(d$getPTime), max(d$relPTime, na.rm=T)) ;  
      testEdges  
[1] "2005-08-23 13:24:32 CDT" "2005-08-24 00:57:16 CDT"  
  
<R13> testTime = diff(testEdges) ; testTime  
Time difference of 11.54556 hours  
  
<R14> prettyEdges = c(testEdges[1]-60*60, testEdges[2]+60*60)
```

### A.2.1 Basics

How many files deliveries were attempted?

```
<R15> nrow(d)  
[1] 40000
```

How many failed on the get end?

```
<R16> sum(is.na(d$getDur))  
[1] 0
```

How many failed on the release end?

```
<R17> sum(is.na(d$relDur))  
[1] 0
```

Merge job and segment numbers

```
<R18> d$jobseg = paste(d$job, d$segment)
```

What were the jobs and segments?

```
<R19> d$jobseg[ is.na(d$relDur) ]  
character(0)  
  
<R20> #var successfulDeliveries = nrow(d)  
40000
```

How come this isn't 40,000?

Look up the maximum file number for each job and segment.

```
<R21> largestFNum = tapply(d$fileNum, d$jobseg, max)
```

How many did not get all 40 files?

```
<R22> length( largestFNum[ largestFNum < 40 ] )  
[1] 0
```

Hmmm.

```
<R23> notDone = largestFNum[ largestFNum < 40 ]
```

```
<R24> notDone[1:3]  
<NA> <NA> <NA>  
NA NA NA
```

Will have to look these up!

### A.2.2 File delivery rate

```
<R25> fileRatePerDay = nrow(d)/(as.numeric(testTime))*24;  
      fileRatePerDay  
[1] 83148.88
```

In a perfect world, what should be the mean wait time?

```
<R26> filesPerSeg = nrow(d) / length(unique(d$jobseg)) ; filesPerSeg  
[1] 40
```

Assume that every segment runs the entire length of the test. Therefore (in minutes),

```
<R27> meanWaitTime = as.numeric(testTime)*60 / filesPerSeg ;  
      meanWaitTime  
[1] 17.31833
```

How many seconds per file?

```
<R28> secondsPerFile = as.numeric(testTime)*60*60 / nrow(d) ;  
      secondsPerFile  
[1] 1.0391
```

### A.2.3 Number of segments running

Let's plot how many segments were running at a given time.

A segment turns on when it does its first "get next file". It turns off when it does its last release.

How many segments are there?

```
<R29> length( unique( d$jobseg ) )  
[1] 1000
```

So we have a record of all 1000 segments. Good!

Segment starts when the first file is requested

```

<R30> segStart = data.frame( time=d$getPTime[d$fileNum==1], adj=1 )

<R31> segEnd = data.frame( time=d$relPTime[d$fileNum==40], adj=-1 )

<R32> segs = rbind(segStart, segEnd)

<R33> segs = segs[ order(segs$time), ]

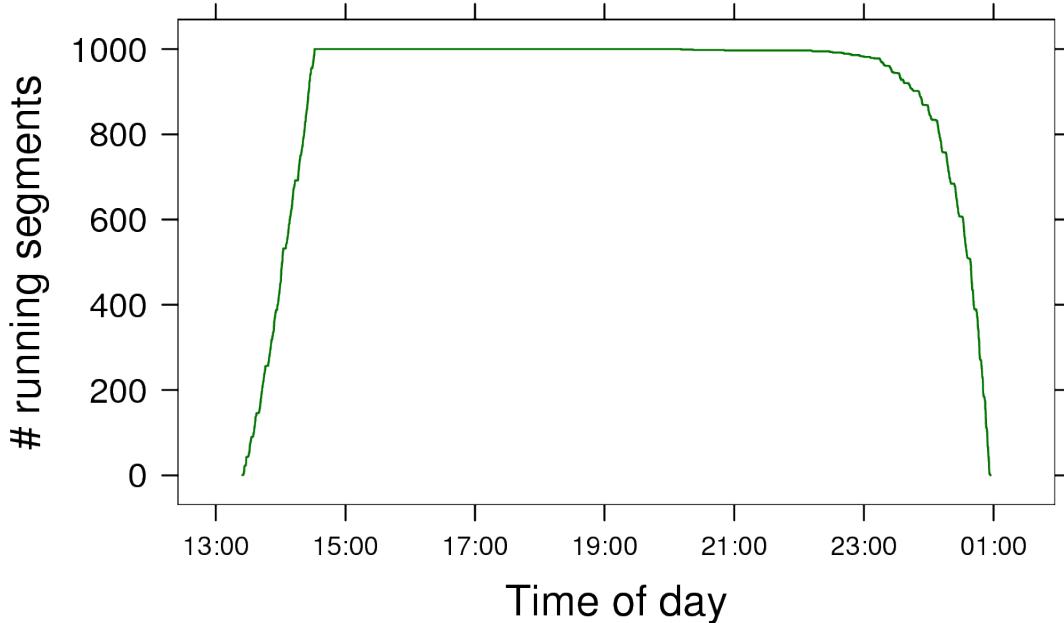
<R34> segs$count = cumsum(segs$adj)

<R35> segs[1:10,]
      time adj count
20 2005-08-23 13:24:32   1     1
160 2005-08-23 13:25:35   1     2
900 2005-08-23 13:25:43   1     3
100 2005-08-23 13:25:49   1     4
60 2005-08-23 13:25:59   1     5
80 2005-08-23 13:26:03   1     6
239 2005-08-23 13:26:04   1     7
680 2005-08-23 13:26:04   1     8
679 2005-08-23 13:26:08   1     9
459 2005-08-23 13:26:20   1    10

<R36> mp(
  xyplot( segs$count ~ segs$time, type="s",
          main="Number of running segments",
          xlab="Time of day",
          ylab="# running segments",
          scales=list(x=list(tick.number=6, cex=0.6)),
          xlim=prettyEdges ),
  "nsegs.png", h=4, w=6 )
#with graphics nsegs.png timeout 120

```

## Number of running segments



### A.2.4 Number of gets and deliveries

Let's count up how many get file requests and deliveries there were per hour

```
<R37> getsPerHourCuts = cut( d$getPTime, "hours" )

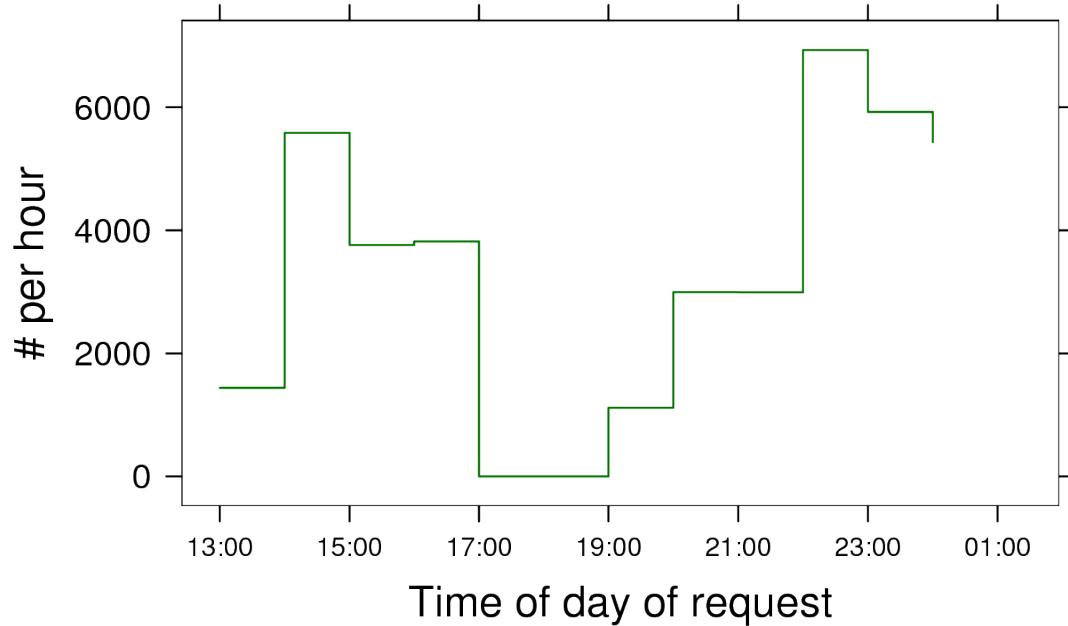
<R38> delsPerHourCuts = cut( d$delPTime, "hours" )

<R39> getsPerHour = tabulate(getsPerHourCuts)

<R40> delsPerHour = tabulate(delsPerHourCuts)

<R41> mp(
  xyplot( getsPerHour ~ as.POSIXct(levels(getsPerHourCuts)),
    main="Get file requests",
    xlab="Time of day of request",
    ylab="# per hour", type="s", xlim=prettyEdges,
    scales=list(x=list(tick.number=6, cex=0.6))
  ),
  "getsPerHour.png", h=4, w=6
)
#with graphics getsPerHour.png timeout 60
```

## Get file requests



```
<R42> mp(
  xyplot( delsPerHour ~ as.POSIXct(levels(delsPerHourCuts)),
    main="File deliveries",
    xlab="Time of day of delivery", xlim=prettyEdges,
    ylab="# per hour", type="s",
    scales=list(x=list(tick.number=6, cex=0.6))
  ),
  "delsPerHour.png", h=4, w=6
)
#with graphics delsPerHour.png timeout 60
```

## File deliveries



Let's just get a cumulative plot of when deliveries occurred.

```
<R43> deliveries = data.frame( time=d$delPTime, adj=1 )
```

```
<R44> deliveries = deliveries[ order(deliveries$time), ]
```

```
<R45> deliveries$count = cumsum(deliveries$adj)
```

```
<R46> nrow(deliveries)
```

```
[1] 40000
```

```
<R47> deliveries[39270:39281,]
```

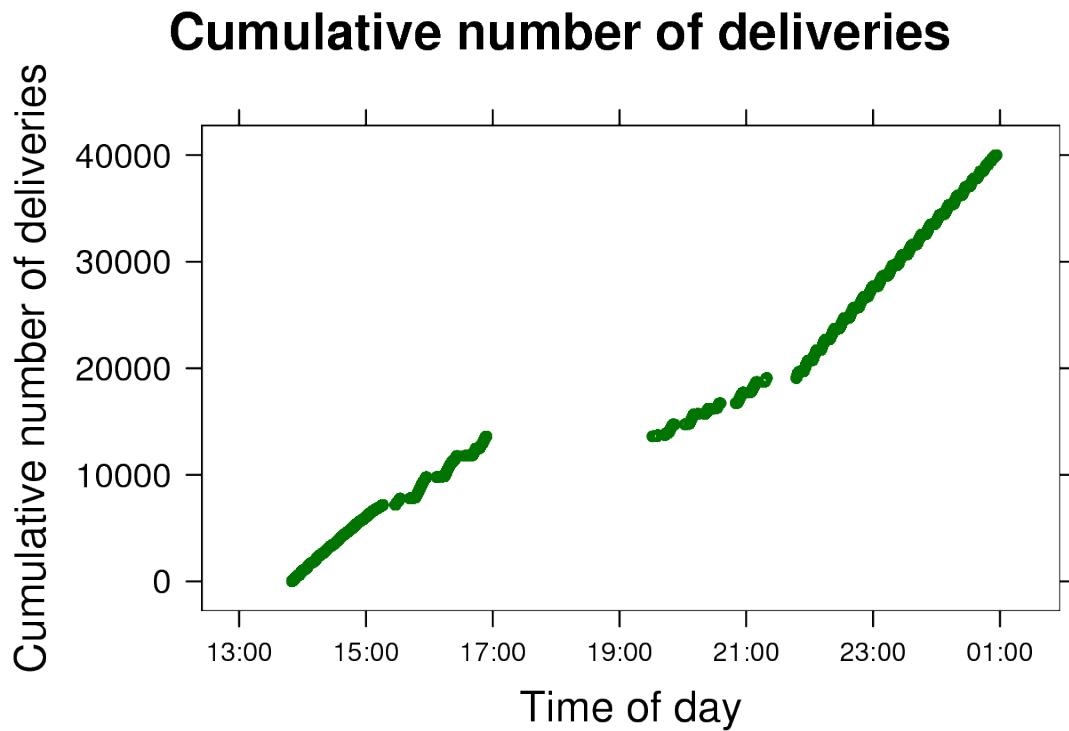
	time	adj	count
--	------	-----	-------

14639	2005-08-24 00:49:42	1	39270
4279	2005-08-24 00:49:42	1	39271
36959	2005-08-24 00:49:42	1	39272
19515	2005-08-24 00:49:43	1	39273
11475	2005-08-24 00:49:43	1	39274
1918	2005-08-24 00:49:43	1	39275
21960	2005-08-24 00:49:44	1	39276
6520	2005-08-24 00:49:44	1	39277
20240	2005-08-24 00:49:44	1	39278
16277	2005-08-24 00:49:44	1	39279
3519	2005-08-24 00:49:44	1	39280
20039	2005-08-24 00:49:44	1	39281

```

<R48> mp(
  xyplot( deliveries$count ~ deliveries$time, type="p",
         main="Cumulative number of deliveries",
         xlab="Time of day",
         ylab="Cumulative number of deliveries",
         scales=list(x=list(tick.number=6, cex=0.6)),
         xlim=prettyEdges ),
  "ndel.png", h=4, w=6 )
#with graphics ndel.png timeout 120

```



There seems to be some significant speed up here! Let's see if we can measure the difference.

```

<R99> deliveries[13000,]
      time adj count
5212 2005-08-23 16:50:26   1 13000

<R105> filesPerSec = function(s, e) {
  nFiles = deliveries$count[e] - deliveries$count[s]
  nSecs = as.numeric(deliveries$time[e]) -
           as.numeric(deliveries$time[s])
  nFiles/nSecs
}

```

Here is the number of seconds per file in the "slow period" (<17:00)

```
<R116> slowP = 1/filesPerSec(1, 13000) ; slowP  
[1] 0.8328795
```

Here is the number of seconds per file in the fast period (>21:50)

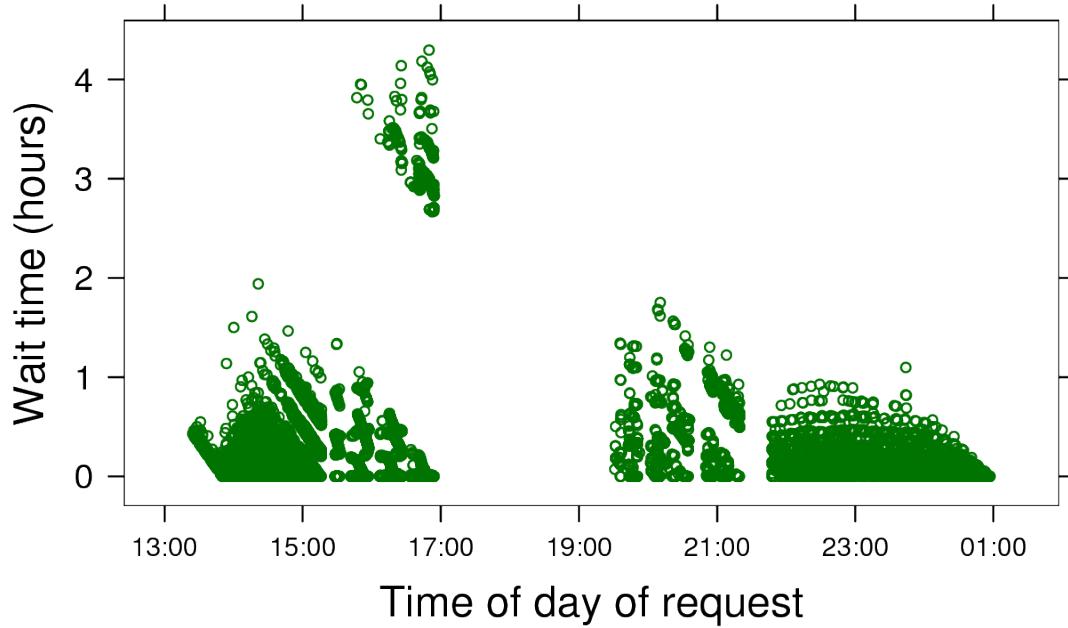
```
<R117> fastP = 1/filesPerSec(20000, 40000) ; fastP  
[1] 0.5445888
```

```
<R123> slowP/fastP  
[1] 1.529373
```

What do the wait times look like?

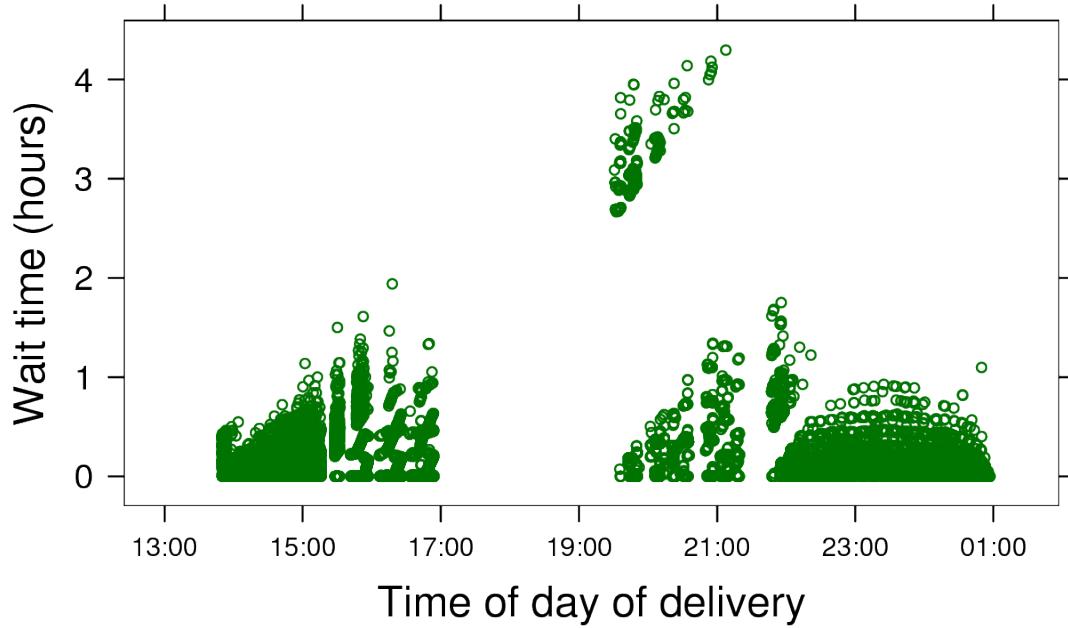
```
<R49> mp(  
  xyplot( getDur/60/60 ~ getPTime, data=d,  
         main="File delivery waits",  
         xlab="Time of day of request", xlim=prettyEdges,  
         ylab="Wait time (hours)",  
         scales=list(x=list(tick.number=6, cex=0.6))  
  ),  
  "getWaits.png", h=4, w=6  
)  
#with graphics getWaits.png timeout 60
```

## File delivery waits



```
<R50> mpC
xyplot( getDur/60/60 ~ delPTime, data=d,
        main="File delivery waits",
        xlab="Time of day of delivery", xlim=prettyEdges,
        ylab="Wait time (hours)",
        scales=list(x=list(tick.number=6, cex=0.6))
),
"delWaits.png", h=4, w=6
)
#with graphics delWaits.png timeout 60
```

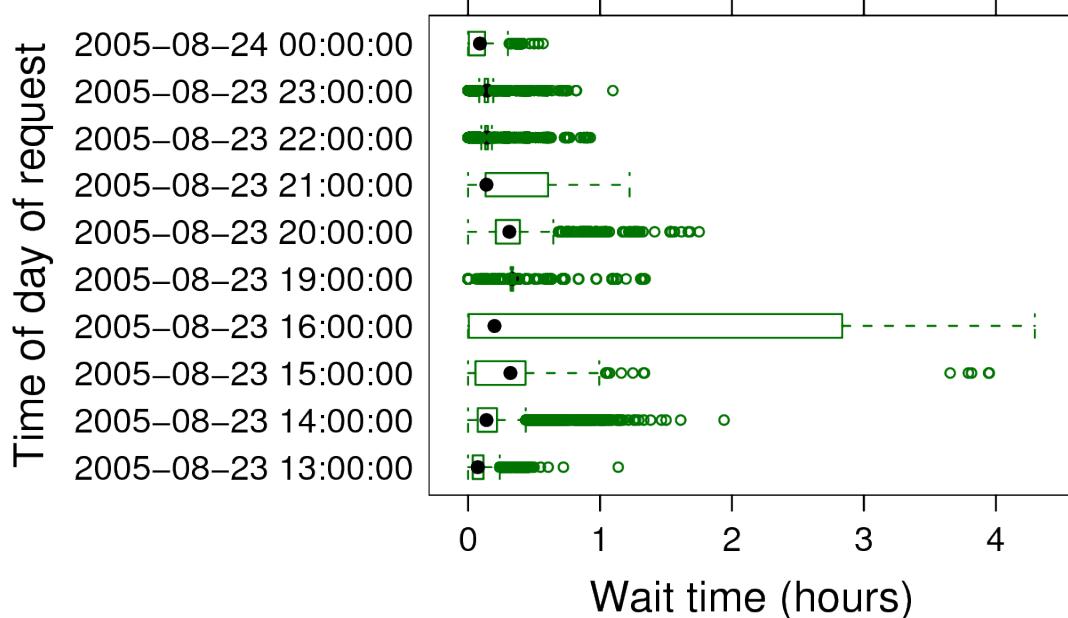
## File delivery waits



These plots make the outliers really stand out and hard to see the mean wait times. Let's try to plot those...

```
<R51> mp(
  bwplot( getsPerHourCuts ~ getDur/60/60, data=d,
         main="File delivery waits",
         xlab="Wait time (hours)", ylab="Time of day of request"
       ),
  "waitsPerHourBw.png", h=4, w=6
)
#with graphics waitsPerHourBw.png timeout 60
```

## File delivery waits



Again, the outliers dominate the plot. Let's just plot medians and how many are over an hour...

```
<R69> cleanNA = function(x) x[ ! is.na(x) ]  
  
<R64> waitMedians = tapply(d$getDur/60, getsPerHourCuts, median,  
    na.rm=T)  
  
<R66> waitMedians = waitMedians[ ! is.na(waitMedians) ]  
  
<R70> waitMeans = tapply(d$getDur/60, getsPerHourCuts, mean) ;  
    waitMeans = cleanNA(waitMeans)  
  
<R75> nOverHour = tapply(d$getDur/60/60 >= 1, getsPerHourCuts, sum) ;  
    nOverHour = cleanNA(nOverHour)  
  
<R77> getsPerHour = getsPerHour[ getsPerHour > 0 ]  
  
<R78> percOverHour = nOverHour / getsPerHour * 100  
  
<R79> nOverHalfHour = tapply(d$getDur/60/60 >= 0.5, getsPerHourCuts,  
    sum) ; nOverHalfHour = cleanNA( nOverHalfHour)
```

```
<R80> percOverHalfHour = nOverHalfHour / getsPerHour * 100

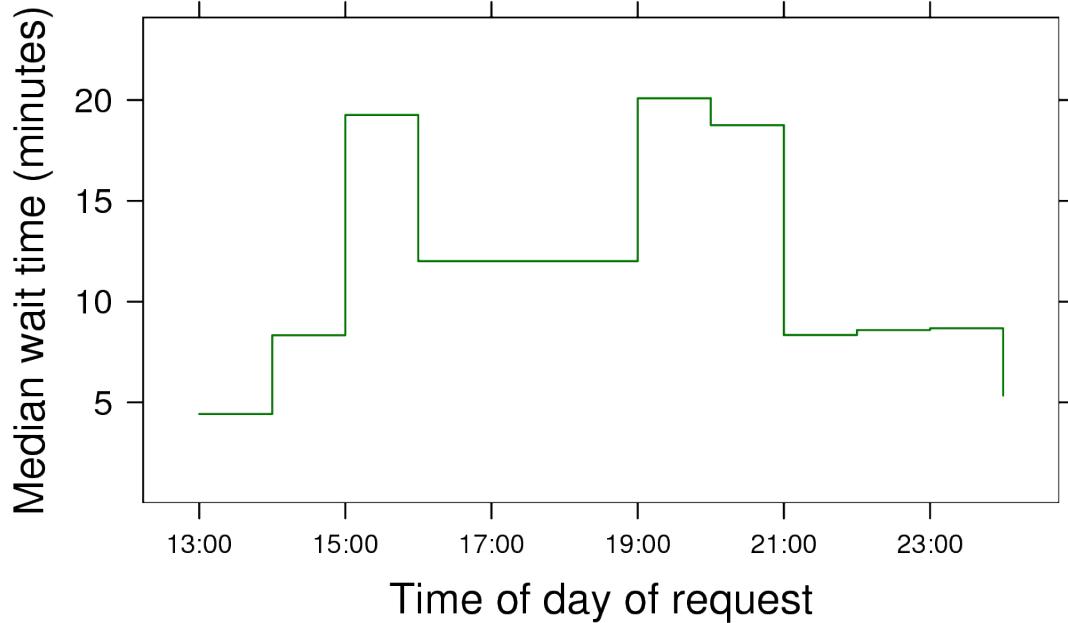
<R81> nUnderMinute = tapply(d$getDur/60 < 1, getsPerHourCuts, sum) ;
      nUnderMinute = cleanNA( nUnderMinute)

<R82> percUnderMinute = nUnderMinute / getsPerHour * 100
```

Let's plot these things

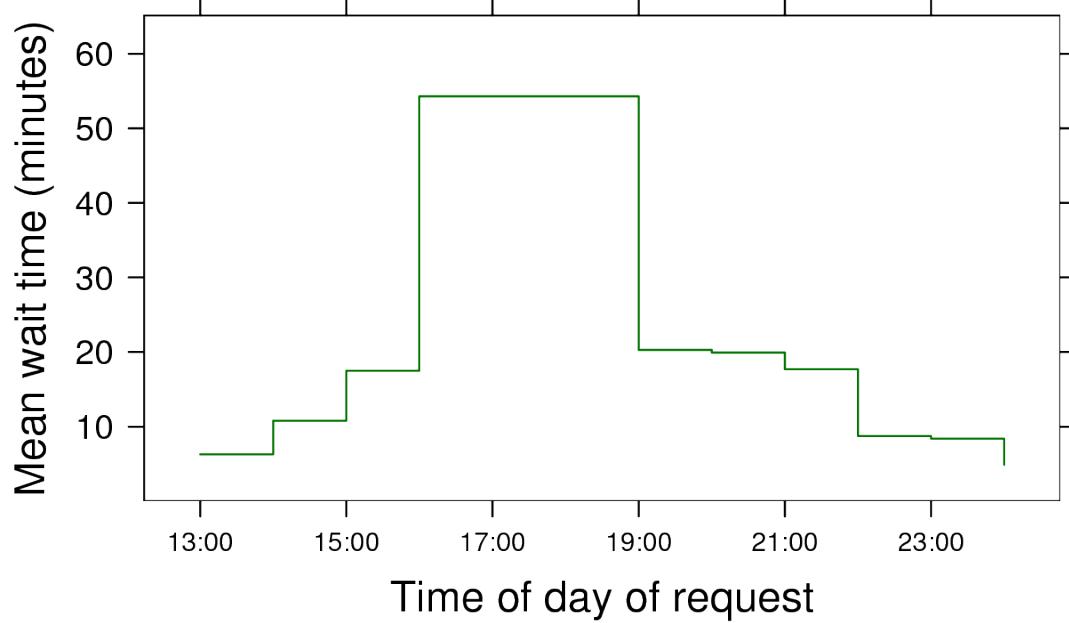
```
<R84> mp(
  xyplot( waitMedians ~ as.POSIXct(names(waitMedians)), type="s",
         main="Median wait time", xlab="Time of day of request",
         ylab="Median wait time (minutes)",
         ylim=c(0, max(waitMedians)*1.2),
         scales=list(x=list(tick.number=6, cex=0.6)))
),
"medians.png", h=4, w=6)
#with graphics medians.png timeout 60
```

## Median wait time



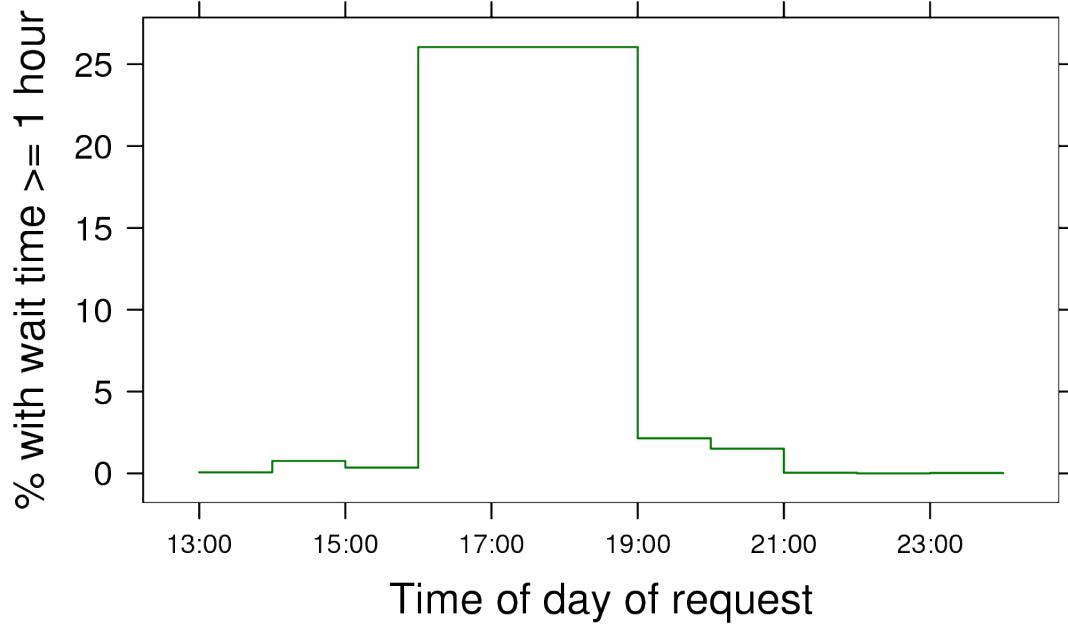
```
<R85> mp(
  xyplot( waitMeans ~ as.POSIXct(names(waitMeans)), type="s",
         main="Mean wait time", xlab="Time of day of request",
         ylab="Mean wait time (minutes)",
         ylim=c(0, max(waitMeans)*1.2),
         scales=list(x=list(tick.number=6, cex=0.6))
  ),
  "means.png", h=4, w=6)
#with graphics means.png timeout 60
```

## Mean wait time

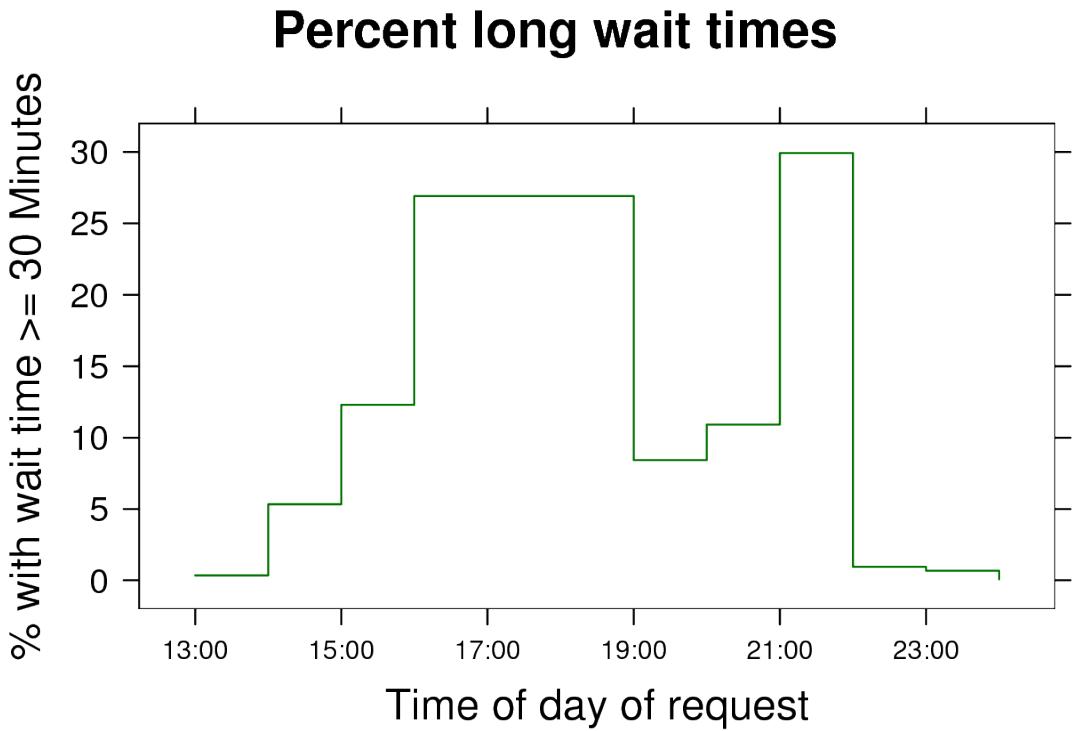


```
<R86> mp(
  xyplot( percOverHour ~ as.POSIXct(names(percOverHour)), type="s",
         main="Percent long wait times", xlab="Time of day of
request",
         ylab="% with wait time >= 1 hour",
         scales=list(x=list(tick.number=6, cex=0.6))
  ),
  "overHour.png", h=4, w=6)
#with graphics overHour.png timeout 60
```

## Percent long wait times



```
<R87> mp(
  xyplot( percOverHalfHour ~ as.POSIXct(names(percOverHalfHour)),
  type="s",
    main="Percent long wait times", xlab="Time of day of
request",
    ylab="% with wait time >= 30 Minutes",
    scales=list(x=list(tick.number=6, cex=0.6))
),
"overHalfHour.png", h=4, w=6)
#with graphics overHalfHour.png timeout 60
```



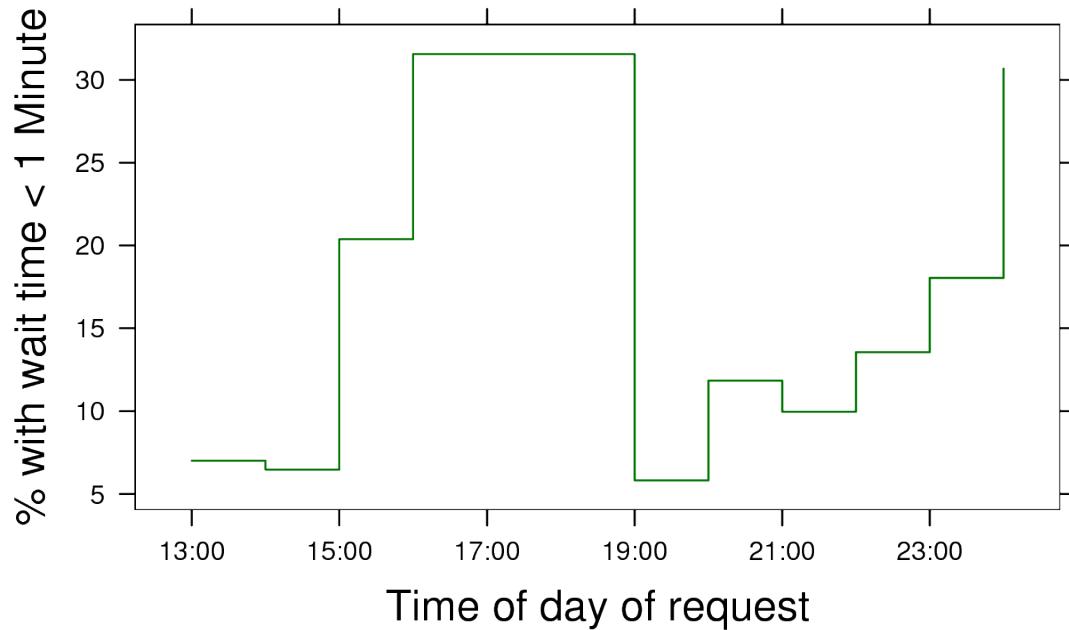
Look for short wait times.

```

<R88> mp(
  xyplot( percUnderMinute ~ as.POSIXct(names(percUnderMinute)),
  type="s",
    main="Percent short wait times", xlab="Time of day of
request",
    ylab="% with wait time < 1 Minute",
    scales=list(tick.number=6, cex=0.6)
  ),
  "underMinute.png", h=4, w=6)
#with graphics underMinute.png timeout 60

```

## Percent short wait times



### A.2.5 Look at number of open requests

Let's look at the number of instantaneous requests that are open at any given time.

Get next file requests...

```
<R89> gnf = data.frame( time=d$getPTime, adj=1 )
```

File deliveries (request fulfilled)

```
<R90> gnc = data.frame( time=d$delPTime, adj=-1 )
```

Join them,

```
<R91> gn = rbind(gnf, gnc)
```

Sort by time,

```
<R92> gn = gn[ order(gn$time), ]
```

Do a running count of # of unfulfilled get next file requests

```
<R93> gn$count = cumsum(gn$adj)
```

```
<R94> gn[1:10,]
```

```

          time adj count
761 2005-08-23 13:24:32 1 1
6361 2005-08-23 13:25:35 1 2
35961 2005-08-23 13:25:43 1 3
3961 2005-08-23 13:25:49 1 4
2361 2005-08-23 13:25:59 1 5
3161 2005-08-23 13:26:03 1 6
9521 2005-08-23 13:26:04 1 7
27161 2005-08-23 13:26:04 1 8
27121 2005-08-23 13:26:08 1 9
18321 2005-08-23 13:26:20 1 10

<R95> summary(gn$count)
   Min. 1st Qu. Median      Mean 3rd Qu.      Max.
0.0    833.0  934.0  840.5  955.0  1000.0

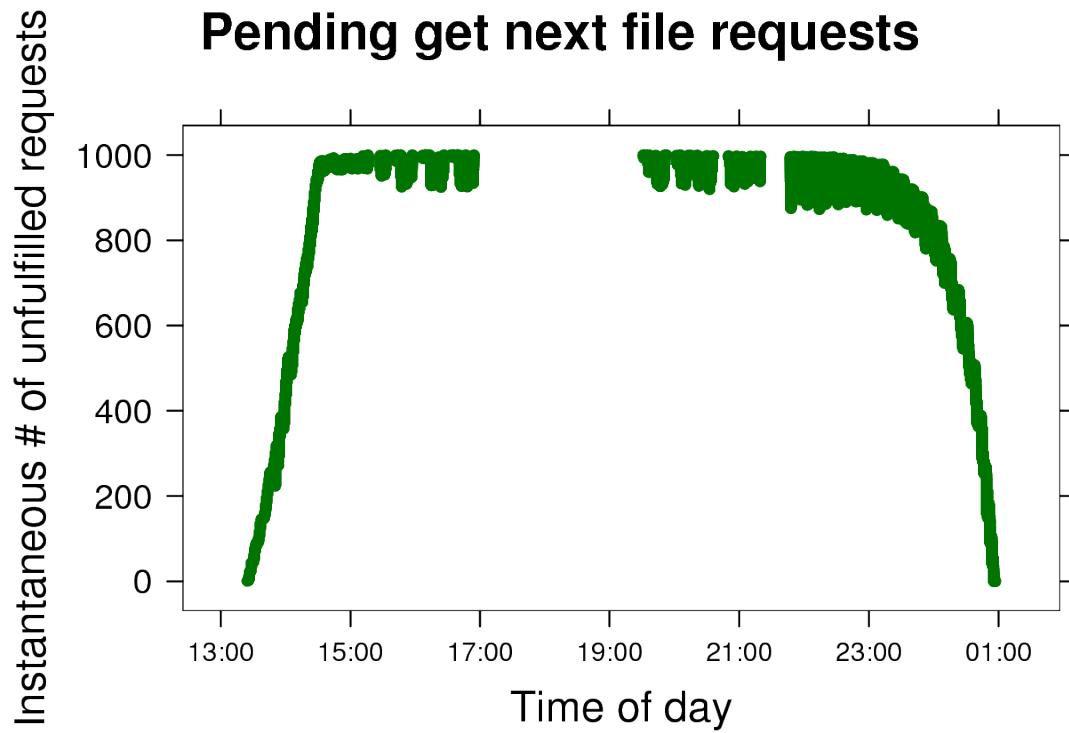
```

Plot it...

```

<R96> mp(
  xyplot( gn$count ~ gn$time, type="p",
          main="Pending get next file requests",
          xlab="Time of day",
          ylab="Instantaneous # of unfulfilled requests",
          scales=list(x=list(tick.number=6, cex=0.6)),
          xlim=prettyEdges, pex=0.1 ),
  "unfulfilled.png", h=4, w=6 )
#with graphics unfulfilled.png timeout 240

```



I think this just shows the affect of the request wait time being longer than the request rate.